Distributed Mini-Batch SDCA

Distributed Mini-Batch SDCA

Martin Takáč
Lehigh University
Bethlehem, PA, USA
Takac.MT@gmail.com

Peter Richtárik
University of Edinburgh
Edinburgh, UK
Peter.Richtarik@ed.ac.uk

Nathan Srebro
Toyota Technological Institute at Chicago
Chicago, IL, USA
nati@ttic.edu

Abstract

We present an improved analysis of mini-batched stochastic dual coordinate ascent for regularized empirical loss minimization (i.e. SVM and SVM-type objectives). Our analysis allows for flexible sampling schemes, including where data is distribute across machines, and combines a dependence on the smoothness of the loss and/or the data spread (measured through the spectral norm).

1 Introduction

Stochastic optimization approaches have significant theoretical and empirical advantages in training linear Support Vector Machines (SVMs) and other regularized loss minimization problems, and are often the methods of choice in practice. Such methods use a single, randomly chosen, training example at each iteration. In the context of SVMs, many variations of stochastic gradient descent (SGD) have been suggested, based on primal stochastic gradients (e.g. Pegasos [25], NORMA [30], SAG [22], MISO [15], S2GD [11], mS2GD [10] and Prox-SVRG [9, 17]). In this paper we focus on SDCA—stochastic dual coordinate ascent—which is based on improvements to the dual problem, again considering only a single randomly chosen training example, and thus only a single randomly chosen dual variable, at each iteration [7, 21, 13]. Especially when accurate solutions are desired, SDCA has better complexity guarantee, and often performs better in practice than SGD [7, 23].

The inherent sequential nature of such approaches becomes a problematic limitation in parallel and distributed settings as the predictor must be updated after each training point is processed, providing very little opportunity for parallelization. A popular remedy is to use *mini-batches*: the use several training points at each iteration, calculating the update based on each point separately and aggregating the updates. The question is then whether basing each iteration on several points can indeed reduce the number of required iterations, and thus yield parallelization speedups.

For SGD with a non-smooth loss, mini-batching does not reduce the number of worst-case required iterations and thus does not allow parallel speedups in the worst case. However, when the loss function is smooth, mini-batching can be beneficial and linear speedups can be obtained, even when the mini-batch sizes scales polynomially with the total training set size [4, 1, 3]. Alternatively, even for non-smooth loss, linear speedups can also be ensured if the data is reasonably well-spread, as measured by the spectral norm of the data, as long as the mini-batch size is not larger then the inverse of this spectral norm [27].

For SDCA, using a mini-batch corresponds to updating multiple coordinates concurrently and independently. If appropriate care is taken with the updates (see Section 6), then using a mini-batch size as large as the inverse spectral norm leads to a reduction in the number of iterations, and allows

linear parallelization speedups, even when the loss is non-smooth [2, 27]. This parallels the SGD mini-batch analysis for non-smooth loss. But can mini-batching also be beneficial for SDCA with smooth losses and without a data-spread (spectral norm) assumptions, as with SGD? What mini-batch sizes allow for parallel speedups? In this paper we answer these questions and show that as with SGD, when the loss function is smooth, using mini-batches with SDCA yields a linear reduction in the number of iterations and thus allows for linear parallelization speedups, up to similar polynomial limits on the mini-batch size. Furthermore, we provide an analysis that combines the benefits of smoothness with the data-dependent benefits of a low spectral norm, and thus allows for even large mini-batch sizes when the loss is smooth *and* the data is well-spread.

Another issue that we address is the way mini-batches are sampled. Straight-forward mini-batch analysis, including previous analysis of mini-batch SDCA [27], assume that at each iteration we pick a mini-batch of size b uniformly at random from among all subsets of b training examples. In practice, though, data is often partitioned between $C \leq b$ machines, and at each iteration b/C points are samples from each machine, yielding a mini-batch that is not uniformly distributed among all possible subsets (e.g. we have zero probability of using b points from the same machine as a mini-batch). Other architectural restrictions might lead to different sampling schemes. The analysis we present can be easily applied to different sampling schemes, and in particular we consider distributed sampling as described above and show that essentially the same guarantees (with minor modification) hold also for this more realistic sampling scheme.

Finally, we compare our optimization guarantees to those recently established for CoCoA+ [14]. CoCoA+ is an alternative dual-based distributed optimization approach, which can be viewed as including mini-batch SDCA as a special case, and going beyond SDCA to potentially more powerful optimization. At each iteration of CoCoA+, several groups of dual variables are updated. We focus on CoCoA+SDCA, where each group is updated using some number of SDCA iterations. When each group consists of a single variable, this reduces exactly to mini-batch SDCA. Allowing for multiple SDCA iterations on larger groups of variables yields a method that is more computationally demanding that mini-batch SDCA, and intuitively should be better than SDCA (and does appear better in practice). However, we show that our mini-batch SDCA analysis strictly dominates the CoCoA+ analysis: that is, with the same number of total dual variables updated per iteration, and thus less computation, our mini-batch SDCA guarantees are strictly better than those obtained for CoCoA+. Mini-batch SDCA is thus a simpler, computationally cheaper method, with better *guarantees* than those established for CoCoA+.

Although SDCA is a dual-method, improving the dual at each iteration, following the analysis methodology of [23], all our guarantees are on the duality gap, and thus on the *primal* sub-optimality, that is on the actual regularized error we care about.

2 Setup and Preliminaries

We consider the problem of minimizing the regularized empirical loss

$$\min_{w\in\mathbb{R}^d} \mathcal{P}(w) := \tfrac{1}{n}\textstyle\sum_{i=1}^n \phi_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2, \tag{P}$$

where $x_1, \ldots, x_n \in \mathbb{R}^d$ are given training examples, $\lambda > 0$ is a given regularization parameter and $\phi_i : \mathbb{R} \to \mathbb{R}$ are given non-negative convex loss functions that already incorporate the labels (e.g. $\phi_i(z) = \phi(y_i z)$ where $y_i \in \pm 1$ are given labels). Instead of solving (P), we solve the dual [23]

$$\max_{\alpha\in\mathbb{R}^n} \mathcal{D}(\alpha) := -\tfrac{1}{n}\textstyle\sum_{i=1}^n \phi_i^*(-\alpha_i) - \frac{\lambda}{2}\|\frac{1}{\lambda n} X^T\alpha\|_2^2, \tag{D}$$

where $\phi_i^*(u) : \mathbb{R} \to \mathbb{R}$ is the convex conjugate of ϕ_i defined in the standard way as $\phi_i^*(u) = \max_z(zu - \phi(z))$ and $X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n\times d}$ is the data matrix, where each row corresponds to one sample and each column corresponds to one feature. If α^* is a dual-optimum of (D) then $w^* = \frac{1}{\lambda n} X^T \alpha^*$ is a primal-optimum of (P). We therefor consider the mapping $w_\alpha = \frac{1}{\lambda n} X^T \alpha$ and define the *duality gap* of a feasible $\alpha \in \mathrm{dom}(\mathcal{D})$ as:

$$\mathcal{G}(\alpha) := \mathcal{P}(w_\alpha) - \mathcal{D}(\alpha). \tag{G}$$

Stochastic Dual Coordinate Ascent (SDCA) SDCA is a coordinate ascent algorithm optimizing the dual (D). At t-th iteration of SCDA a coordinate $i \in \langle n\rangle := \{1, 2, \ldots, n\}$ is chosen at random

and then a new iteration is obtained by updating only the i-th coordinate and keeping all other coordinates of α unchanged i.e. $\alpha^{(t+1)} = \alpha^{(t)} + \Delta\alpha_i^{(t)} e_i$, where

$$\Delta\alpha_i^{(t)} = \arg\min_{\delta \in \mathbb{R}} \mathcal{D}(\alpha^{(t)} + \delta e_i). \tag{1}$$

Assumptions on Loss Function We analyze mini-batched SDCA under one of two different assumptions on the loss functions: that they are L-Lipschitz continuous (but potentially non-smooth), or that they are $(1/\gamma)$-smooth. Formally: i) L-**Lipschitz continuous loss:** $\forall i, \forall a, b \in \mathbb{R}$ we have $|\phi_i(a) - \phi_i(b)| \leq L|a - b|$, ii) $(1/\gamma)$-**smooth loss:** Each loss function ϕ_i if differentiable and its derivative is $(1/\gamma)$-Lipschitz continuous, i.e. $\forall a, b \in \mathbb{R}$ we have $|\phi_i'(a) - \phi_i'(b)| \leq \frac{1}{\gamma}|a - b|$; iii) We also assume ϕ_i are non-negative and that $\phi_i(0) \leq 1$ for all i.

For a positive vector $v = (v_1, \ldots, v_n)^T > 0$ we define a weighted Euclidean norm $\|\alpha\|_v^2 = \sum_{i=1}^n v_i \alpha_i^2$. Instead of assuming the data is uniformly bounded, we will frequently refer to the weighted norm on $\mathbb{R}^n$ with weights proportional to the squared magnitudes, i.e. $v_i \sim \|x_i\|^2$.

3 Mini-Batched SDCA

At each iteration of mini-batched SDCA, a subset $S \subseteq \langle n \rangle$ of the coordinates is chosen at random (see below for a discussion of the sampling distribution) and a new dual iterate is obtained by independently updating only the chosen coordinates. Since each coordinate is updated independently, mini-batch SDCA is amenable to parallelization.

The naïve approach is to use the same update rule for each coordinate as in serial case: the update is then given by $\alpha^{(t+1)} = \alpha^{(t)} + \sum_{i \in S} \Delta\alpha_i^{(t)}$ where $\Delta\alpha_i^{(t)}$ is given by (1). Such a naïve approach could be fine if the mini-batch size is very small and the data is "spread-out" enough [2]. However, more generally, not only might such a mini-batch iteration not be better than an iteration based on only a single point, but such a naïve mini-batch update might actually be much worse. In particular, it is easy to construct an example with just two examples where a naïve mini-batch approach will never reach the optimum solution, and diverging behavior frequently occurs in practice on real data sets [27]. The problem here is that the independent updates on multiple similar points might combine together to "overshoot" the optimum and hurt the objective.

An alternative that avoids this problem is to average the updates instead of adding them up, $\alpha^{(t+1)} = \alpha^{(t)} + \frac{1}{|S|} \sum_{i \in S} \Delta\alpha_i^{(t)}$ [8, 29, 28], but such an update is overly conservative: it is not any better than just updating a single dual variable, and cannot lead to parallelization speedups. Following [27], the approach we consider here is to use a summed update $\alpha^{(t+1)} = \alpha^{(t)} + \sum_{i \in S} \Delta\alpha_i^{(t)}$, where the independent updates $\Delta\alpha_i^{(t)}$ are derived from a relaxation of the dual:

$$\Delta\alpha_i^{(t)} = \arg\max_{\delta} -\phi_i^*(-\alpha_i - \delta) + \tfrac{1}{2\lambda n} v_i \delta^2 - w_\alpha^T x_i \delta \tag{2}$$

When $v_i = \|x_i\|^2$, the update exactly agrees with the dual-optimizing update (1). But as we shall see, when larger mini-batches are used, larger values of v_i are required, resulting in smaller steps. The update (2) generalizes [27] where a single parameter $v_i = v$ was used—here we allow v_i to vary between dual variables, accommodating differences in $\|x_i\|$.

To summarize, the mini-batch SDCA algorithm we consider takes as input data X, loss functions ϕ_i, a distribution over subsets $S \subseteq \langle n \rangle$, which we will refer to as the random sampling $\hat{S}$, and a weight vector v, and proceeds as shown on Algorithm 1.

We will refer to several sampling distributions $\hat{S}$, yielding different variants of mini-batch SDCA:
Serial SDCA. $\hat{S}$ is a uniform distribution over singletons. That is, S_t contains a single coordinate chosen uniformly at randomly. Setting $v_i = \|x_i\|$ yields standard SDCA.
Standard Mini-batch SDCA. $\hat{S}$ is a uniform distribution over subsets of size b. **Distributed SDCA.** Consider a setting with C machines, n total data points and a mini-batch size b, where for simplicity n and b are both integer multiples of C. For a partition of the n coordinates into C equal sized subsets $\{P_c\}_{c=1}^C$, consider the following sampling distribution $\hat{S}$: for each $c = 1..C$, choose a

Algorithm 1 mSDCA: minibatch Stochastic Dual Coordinate Ascent

Input: $X, y, \hat{S}, v$
set $\alpha^{(0)} = \mathbf{0} \in \mathbb{R}^n$
for $t = 0, 1, 2, \ldots$ **do**
  choose S_t according the distribution $\hat{S}$
  set $\alpha^{(t+1)} = \alpha^{(t)}$; $w_\alpha = \frac{1}{\lambda n} X^T \alpha^{(t)}$
  for $i \in S_t$ **in parallel do**
    $\Delta\alpha_i^{(t)} = \arg\max_\delta -\phi_i^*(-\alpha_i - \delta) - \frac{1}{2\lambda n} v_i \delta^2 - w_\alpha^T x_i \delta$
    $\alpha_i^{(t+1)} = \alpha_i^{(t)} + \Delta\alpha_i^{(t)}$
  end for
end for

subset $S^c \subset P_c$ uniformly and independently at random among all such subsets of size b/C, and then take their union. We refer to such a sample as a (C, b)-distributed sampling. Such a sampling is suitable in a distributed environment when n samples are equally partitioned over C computational nodes in a cluster [19, 16]. When $C = 1$ we obtain the Standard Mini-batch sampling.

The main question we now need to address is what weights v_i are suitable for use with each of the above sampling schemes, and what optimization guarantee to they yield. To answer this question, in the next Section we will introduce the notion of Expected Separable Overapproximations.

4 Expected Separable Overapproximation

In this Section we will make use of the Expected Separable Overapproximation (ESO) theory introduced in [20] and further extended e.g. in [16, 19, 18].

4.1 Motivation

Consider the t-th iteration of mini-batch SDCA. Our current iterate is $\alpha^{(t)}$ and we have chosen a set S_t of coordinates which we will update in current iteration. We need to compute the updates to those coordinates, i.e. $\forall i \in S_t$ we need to compute $\Delta\alpha_i^{(t)}$. Maybe the natural way how to define the updates would be to define them such that $D(\alpha^{(t+1)})$ is as large as possible, i.e. that we maximize $D(\alpha^{(t)} + \sum_{i \in S_t} \Delta\alpha_i^{(t)} e_i)$. However, this e.g. for hinge loss would lead to a QP, hence the computation cost would be substantial. The main disadvantage of this approach is the fact that the updates for different coordinates are dependent on each other, i.e. the value of $\Delta\alpha_i^{(t)}$ depends on all coordinates in S_t. This make it hard to parallelize. Considering the fact that S_t is a random set, maybe one would like to define the updates so that the updates doesn't depend on current choice of S_t and that they maximize the expected value of D at next iteration. In this case we are facing following maximization problem

$$\max_{t \in \mathbb{R}^n} \mathbb{E}[D(\alpha^{(t)} + t_{[S_t]})], \tag{3}$$

where $t_{[S_t]}$ is a masking operator setting all coordinates of t which are not in set S_t to zero, i.e. $(t_{[S_t]})_i = t_i$ if $i \in S_t$ and $(t_{[S_t]})_i = 0$ otherwise. The expectation in (3) is considered over the distribution $\hat{S}$. After we get the optimal solution t^* of (3) we can define $\Delta\alpha_i^{(t)} = t_i^*$ for all $i \in S_t$. Therefore $\alpha^{(t+1)} = \alpha^{(t)} + \sum_{i \in S_t} \Delta\alpha_i^{(t)} = \alpha^{(t)} + t_{[S_t]}^*$. However, now the problem (3) is even more complicated. The remedy is to replace $\mathbb{E}[D(\alpha^{(t)} + t_{[S_t]})]$ by its separable lowerbound. Then due to the fact that it will be separable, the update for any coordinate i will be independent on the other coordinates in S_t and moreover, the updates will be obtained by solving 1D problem.

4.2 Lower-bound

Let us first state the definition of ESO.

Definition 1 (Expected Separable Overapproximation [20])**.** Assume that sampling $\hat{S}$ has uniform marginals. Then we say that function f admits v-ESO with respect to the sampling $\hat{S}$ if $\forall x, t \in \mathbb{R}^n$ we have

$$\mathbb{E}[f(\alpha + t_{[\hat{S}]})] \leq f(\alpha) + \frac{\mathbb{E}[|\hat{S}|]}{n}(\langle \nabla f(\alpha), t\rangle + \frac{1}{2}\|t\|_v^2). \tag{4}$$

Let us now just assume that we can find such a vector v such that (4) holds (we show how to find v in Section 4.3) and we now show how to derive the lowerbound of $\mathbb{E}[D(\alpha^{(t)} + t_{[S_t]})]$. If we write (4) for a particular choice of f, namely for $f(\alpha) = \|\frac{1}{\lambda n}X^T\alpha\|_2^2$ we obtain

$$\mathbb{E}[\|\frac{1}{\lambda n}X^T(\alpha + t_{[\hat{S}]})\|_2^2] \overset{(4)}{\leq} \|w_\alpha\|^2 + \frac{\mathbb{E}[|\hat{S}|]}{n}(\|\frac{1}{\lambda n}t\|_v^2 + \frac{2}{\lambda n}t^T X w_\alpha). \tag{5}$$

Now we can derive the expected lowerbound of $\mathcal{D}$ as follows

$$\begin{aligned}\mathbb{E}[\mathcal{D}(\alpha + t_{[\hat{S}]})] &\overset{(D)}{=} \mathbb{E}[-\frac{1}{n}\textstyle\sum_{i=1}^n \phi_i^*(-(\alpha + t_{[\hat{S}]})_i)] - \mathbb{E}[\frac{\lambda}{2}\|\frac{1}{\lambda n}X^T(\alpha + t_{[\hat{S}]})\|_2^2] \\ &\overset{(5)}{\geq} -\frac{\lambda}{2}\|w_\alpha\|^2 - \frac{\mathbb{E}[|\hat{S}|]}{n}\frac{1}{n}\textstyle\sum_{i=1}^n \phi_i^*(-\alpha_i - t_i) - (1 - \frac{\mathbb{E}[|\hat{S}|]}{n})\frac{1}{n}\sum_{i=1}^n \phi_i^*(-\alpha_i) \\ &\quad - \frac{\mathbb{E}[|\hat{S}|]}{n}\frac{\lambda}{2}(\|\frac{1}{\lambda n}t\|_v^2 + \frac{2}{\lambda n}t^T X w_\alpha),\end{aligned} \tag{6}$$

where in the first inequality for the first part we have used the fact that the function is separable (see Theorem 4 in [20]). If we define

$$\mathcal{H}(t, \alpha) := -\frac{1}{n}\textstyle\sum_{i=1}^n \phi_i^*(-(\alpha_i + t_i)) - \frac{\lambda}{2}\|w_\alpha\|^2 - \frac{\lambda}{2}\|\frac{1}{\lambda n}t\|_v^2 - \frac{1}{n}t^T X w_\alpha, \tag{7}$$

then it is easy to see that we can find a separable (in t) expected lower approximation of $\mathcal{D}$, i.e. it holds $\forall \alpha, t \in \mathbb{R}^n$ that $\mathbb{E}[\mathcal{D}(\alpha + t_{[\hat{S}]})] \overset{(6)}{\geq} \frac{b}{n}\mathcal{H}(t, \alpha) + \left(1 - \frac{b}{n}\right)\mathcal{D}(\alpha)$, where $b := \mathbb{E}[|\hat{S}|]$ is the average number of mini-batch. Now let us note again that it is very hard to maximize $\mathbb{E}\left[\mathcal{D}\left(\alpha + t_{[\hat{S}]}\right)\right]$ in t, but maximize of $\mathcal{H}(t, \alpha)$ in t is very simple, because this function is simple and separable in t. It is also easy to verify that the steps in Algorithm 1 are maximizing $\mathcal{H}$.

4.3 Computing ESO Parameter

In previous Section we have shown that using ESO we can find a separable lowerbound of $\mathbb{E}[\mathcal{D}(\alpha + t_{[\hat{S}]})]$. However, we haven't explained how the ESO parameter (vector v) can be obtained.

In this Section we present some of the results obtained in literature [20, 6, 16, 5] for formulas for computing vector v for samplings described in Section 3. Let us mention that all formulas are **data dependent**. Some of them involves the spectral radius of following matrix $D^{-\frac{1}{2}}XX^TD^{-\frac{1}{2}}$, where $D = \mathbf{diag}(XX^T)$ which we will denote by σ^2, hence $\sigma^2 := \max_{\alpha \in \mathbb{R}^n : \|\alpha\|=1} \frac{1}{n}\|X^T D^{-\frac{1}{2}}\alpha\|^2$. Note that this can be in practise impossible to compute (we can estimate is using e.g. power method) or we can use an upper-bound (derived in Lemma 5.4 [5]) by $\omega = \max_{i \in \langle n \rangle} \frac{1}{n}\frac{\sum_{j=1}^d \|x_i\|_0 (x_i^T e_j)^2}{\sum_{j=1}^d (x_i^T e_j)^2}$, where by $\|x_i\|_0$ we have denoted a number of non-zero elements of i-th data point.

Serial SDCA. In this simplest case we can define $v_i = \|x_i\|^2$.

Standard Mini-batch SDCA. In standard mini-batch we can choose $v_i = (1 + \frac{(b-1)(n\sigma^2 - 1)}{\max\{1, n-1\}})\|x_i\|^2$. If the data matrix X is sparse, we can define $v_i = \sum_{j=1}^d (x_i^T e_j)^2 (1 + \frac{(b-1)(\|x_i\|_0 - 1)}{n-1})$.

Distributed SDCA. In distributed case we can choose $v_i = \frac{b}{b-C}(1 + \frac{(b-C)(n\sigma^2 - 1)}{\max\{C, n-C\}})\|x_i\|^2$, provided that $b \geq 2C$ and $v_i = (1 + b\sigma^2)\|x_i\|^2$ if $b = C$. A simple upper-bound valid for any b can be derived as follows $v_i = 2(1 + b\sigma^2)\|x_i\|^2$.

5 Convergence Guarantees

We are now ready to present optimization guarantees for Algorithm 1 based on the ESO parameters studied in the previous Section. These theorems extends the serial case of [23] to mini-batch setting. The Theorems are based on weights v are chosen such that $f(\alpha) = \|\frac{1}{\lambda n}X^T\alpha\|^2$ admits v-ESO for a sampling $\hat{S}$ used in the Algorithm 1. Proofs are provided in the supplemental material.

Theorem 2 ($(1/\gamma)$-Smooth Loss)**.** *If the losses are $(1/\gamma)$-smooth and $f(\alpha) = \|\frac{1}{\lambda n} X^T \alpha\|^2$ admits v-ESO for the sampling $\hat{S}$, then for a desired duality gap $\epsilon_{\mathcal{G}} > 0$, using Algorithm 1, if we choose*

$$T \geq \frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\log\left(\frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\frac{1}{\epsilon_{\mathcal{G}}}\right) \tag{8}$$

we have that $\mathbb{E}[\mathcal{P}(w_T) - \mathcal{D}(\alpha_T)] \leq \epsilon_{\mathcal{G}}$. To obtain $\mathbb{E}[\mathcal{P}(\bar{w}) - \mathcal{D}(\bar{\alpha})] \leq \epsilon_{\mathcal{G}}$, it is sufficient to choose $T_0 \geq \frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\log\left(\frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\frac{1}{(T-T_0)\epsilon_{\mathcal{G}}}\right)$, where

$$\bar{\alpha} = \frac{1}{T-T_0}\sum_{t=T_0+1}^{T-1}\alpha^{(t)}. \tag{9}$$

Moreover, if $\tilde{T} \geq \frac{\|v\|_\infty + \lambda n\gamma}{b\lambda\gamma}\log\left(\frac{\|v\|_\infty + \lambda n\gamma}{b\lambda\gamma}\frac{1}{\epsilon_{\mathcal{G}}\rho}\right)$ then $\mathbb{P}(\mathcal{P}(w_{\tilde{T}}) - \mathcal{D}(\alpha_{\tilde{T}}) \leq \epsilon_{\mathcal{G}}) \geq 1 - \rho$.

Theorem 3 (L-Lipschitz Continuous Loss)**.** *If the losses are L-Lipschitz and $f(\alpha) = \|\frac{1}{\lambda n} X^T \alpha\|^2$ admits v-ESO for the sampling $\hat{S}$, then for a desired duality gap $\epsilon_{\mathcal{G}} > 0$, using Algorithm 1, denoting $G = 4L^2\frac{\sum_{i=1}^n v_i}{n}$, if we choose*

$$T_0 \geq t_0 + \frac{1}{b}\left(\frac{4G}{\lambda\epsilon_{\mathcal{G}}} - 2n\right)_+, \qquad T \geq T_0 + \max\{\lceil\tfrac{n}{b}\rceil, \frac{1}{b}\tfrac{G}{\lambda\epsilon_{\mathcal{G}}}\}, \tag{10}$$

$$t_0 \geq \max(0, \lceil\tfrac{n}{b}\log(2\lambda n\epsilon_D^{(0)}/G)\rceil), \tag{11}$$

we have that $\mathbb{E}[\mathcal{P}(\bar{w}) - \mathcal{D}(\bar{\alpha})] \leq \epsilon_{\mathcal{G}}$, where $\bar{\alpha}$ is defined in (9) *Moreover, when $t \geq T_0$, we have dual sub-optimality bound $\mathbb{E}[\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t)})] \leq \frac{1}{2}\epsilon_{\mathcal{G}}$.*

6 Guarantees and Speedups for Specific Sampling Distributions

Theorems 3 and 2 are stated in terms of ESO parameter v. Let us now consider the specific sampling distribution of interest. Assume for simplicity $\|x_i\| \leq 1$, and define

$$\beta_{\text{srl}} = 1 \qquad \beta_{\text{std}} = 1 + \frac{(b-1)(n\sigma^2-1)}{\max\{1,n-1\}} \qquad \beta_{\text{dist}} = \frac{b}{b-C}\left(1 + \frac{(b-C)(n\sigma^2-1)}{\max\{C,n-C\}}\right) \tag{12}$$

for the serial, standard and distributed sampling schemes respectively, with overall mini-batch size b and distribution over C machines. Using the weights $v_i = \beta$, we then have the following obtain the following iteration complexities:

$(1/\gamma)$-**Smooth Loss.** In this case (8) in Theorem 2 becomes $T \geq \frac{\beta}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\beta}\right)\log\left(\frac{\beta}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\beta}\right)\frac{1}{\epsilon_{\mathcal{G}}}\right)$. and hence the iteration complexity is (ignoring logarithmic terms): $\tilde{\mathcal{O}}\left(\frac{n}{b} + \frac{\beta}{b}\frac{1}{\lambda\gamma}\right)$.

L-**Lipschitz Continuous Loss.** Combining equations (10) and (11), and again ignoring logarithmic factors, we get an iteration complexity of:

$$\tilde{\mathcal{O}}\left(\frac{n}{b} + \frac{\beta}{b}\frac{L^2}{\lambda\epsilon_{\mathcal{G}}}\right). \tag{13}$$

Plugging in β_{std} into (13) recovers the previous analysis of Lipschitz loss with standard sampling.

Both the Lipschitz and smooth cases involve two terms: the first term, $\frac{n}{b}$, always displays a linear improvement as we increase the mini-batch size. However, in the second term, we also have a dependence on the data-dependent $1 \leq \beta \leq b$, which depends on the mini-batch size b. We will have a linear improvement in the second term, i.e. potential for linear speedup, as long as $\beta = O(1)$. For standard sampling we have that $\beta \approx 1 + b\sigma^2$, and so we obtain linear speedups as long as $b = O(1/\sigma^2)$, as discussed in (13). We can now also quantify the effect of distributed sampling and see that it is quite negligible and yields almost the same speedups and the same maximum allows mini-batch size as with standard sampling. Note that typically we will have $C \ll b$, as we would like to process multiple example on each machine—otherwise communication costs would overwhelm computational costs [26]. The analysis supports this choice as well as the extreme choice $C = b$.

Focusing on the smooth loss, it is possible to obtain a linear reduction in the iteration complexity (corresponding to linear speedups) for SGD with mini-batch size of up to $\mathcal{O}(\sqrt{n})$ without any data-dependent assumption, that is regardless of the value of β [4, 1, 3]. Is this possible also with SDCA? Indeed, even if we don't account for the data dependent quantity β, since we always have $\beta \leq b$,

then the iteration complexity of SDCA for mini-batch SDCA with smooth loss is: $\mathcal{O}(1/(\lambda\gamma) + n/b)\log(1/\epsilon))$ a larger mini-batch scales the second term (unconditional on any data dependence), and as long as it is the dominant term, we get linear speedups. Now, to get the min-max learning guarantee, we need to set $\lambda = \Theta(1/\sqrt{n})$(see [24]). Plugging this in, we see that we get linear speedups up to a mini-batch of size $\mathcal{O}(\gamma\sqrt{n})$. Unsurprising, this is the same as the mini-batch SGD guarantee. Now, if we do take data-dependence into account, we have $\beta = \mathcal{O}(1 + b\sigma^2)$ (where σ^2 is as defined above). As long as $b < 1/\sigma^2$, we get linear speedps even if the $1/\lambda$ term is dominant, i.e. regardless of the scaling of lambda relative to n. This is good, because in practice, and especially when the expected error is low, the best lambda is often closer to $1/n$ and not $1/\sqrt{n}$. Returning to the worst-case rate and $\lambda = 1/\sqrt{m}$: we now have an allowed mini-batch size of up to $b = \mathcal{O}(\gamma\sqrt{n}/\sigma^2)$ while still getting linear scaling. That is, we can combined the benefits of both smoothness, where we can scale the mini-batch size by $\sqrt{n}$, and the data dependence, to get an additional scaling by $1/\sigma^2$.

7 Comparison with CoCoA+

CoCoA+ [14] is a recently presented framework and analysis for distributed optimization of the dual (D): Data (and hence dual variables) are partitioned among C machines (as in our distributed sampling), defining C subproblems, one for each machine. At each iteration, the set of dual variables of each of the C machines are updated independently, and then communicated and aggregated across machines. Different local updates can be used, and the CoCoA+ analysis depends on how well the update improves the local subproblem. Here we will consider using local SDCA updates in conjunction with CoCoA+: at each iteration, on each of the C machines, b/C dual variables are selected (as in our distributed sampling), and H iterations of SDCA are performed sequentially on these b/C points (in parallel on each of the C machines, and while considering all other dual variables, including all variables on other machines, as fixed).

We will consider for simplicity 1-smooth loss functions and compare the CoCoA+ guarantees on the number of required iterations [14] to the SDCA gurantees we present here, noting also the differences in the amount of computation per iterations. In all our comparisons, the required communication in each iteration of SDCA and CoCoA+ is identical and amounts to a single distributed averaging of vectors in $\mathbb{R}^d$.

Setting $b = C$ and $H = 1$, we exactly recover mini-batch SDCA with a minibatch of size b, and so we would expect the CoCoA+ analysis to yield the same guarantee. However, our guarantee on the number of required iterations in this case is (ignoring log factors) $\tilde{O}\left(\frac{n}{b} + \frac{1}{b\lambda} + \frac{\sigma^2}{\lambda}\right)$ compared to the CoCoA+ guarantee (ignoring log factors): $\tilde{O}\left(\frac{n}{b} + \frac{n\tilde{\sigma}^2}{b\lambda} + \frac{1}{\lambda} + \frac{\tilde{\sigma}^2}{\lambda^2}\right)$, where $\tilde{\sigma}^2 = \max_c \max_{\alpha:\sum_{i\in\mathcal{P}_c}\|\alpha_i x_i\|^2=1}\left(\frac{C}{n}\|\sum_{i\in\mathcal{P}_c}\alpha_i x_i\|\right) \geq \sigma^2 \geq 1/n$. Our guarantee therefore dominates that of CoCoA+: the second term is worse by a factor of $n\tilde{\sigma}^2 > 1$, the third by a factor of $1/\sigma^2 < 1$ and the fourth term in the CoCoA+ bound, can be particularly bad when λ is small (e.g. when $\lambda \propto 1/n$).

Setting $b > C$ and $H = b/C$, both minibatch SDCA and CoCoA+ perform the same number of SDCA updates (same amount of computation) at each iteration, but while minibatch SDCA's updates are entirely independent, each group of H CoCoA+ updates (the H updates on the same machine) are performed sequentially. We would therefore expect CoCoA+'s updates to be better, and therefore require less iterations. Unfortunately, the CoCoA+ analysis does not show this.

To see the deficiency in the CoCoA+ analysis at another extreme, consider the case where $b = n$**,** $1 < C < n$ **and** $H \to \infty$. In this case, each iteration of mini-batch SDCA is actually a full batch of parallel updates (updating each coordinate independently), while for CoCoA+ this corresponds to fully optimizing each group of n/C dual variables using many SDCA updates (and thus much more computation). Still, the CoCoA+ iteration bound here would be $\tilde{O}\left(1 + \frac{\sigma'\tilde{\sigma}^2}{\lambda}\right)$, where $\sigma' = \max_\alpha \frac{1}{C}\frac{\|X^T\alpha\|}{\sum_c\|\sum_{i\in\mathcal{P}_c}x_i\alpha_i\|}$ and so $\sigma'\tilde{\sigma}^2 \geq \sigma^2$, compared to the better mini-batch SDCA bound $\tilde{O}\left(1 + \frac{\sigma^2}{\lambda}\right)$.

Table 1: Basic characteristics of datasets; obtained from libsvm collection [12].

name	# train. samples	# test samples	# features	Sparsity
epsilon	400,000	–	2,000	100%
rcv1	20,242	677,399	47,236	0.15%
news20	15,000	4,996	1,355,191	0.03%
real-sim	72,309	–	20,958	0.24%

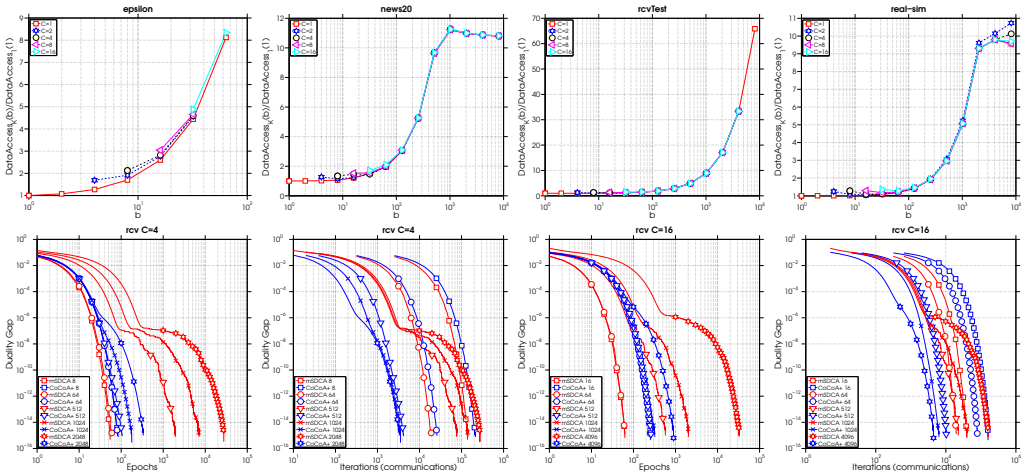

Figure 1: TOP ROW: Number of iterations needed to get an approximate solution is almost the same for standard SDCA and distributed SDCA for $C \in \{1, 2, 4, 8, 16\}$. BOTTOM ROW: Comparison of mSDCA and CoCoA+ when solving the SVM dual problem on $C = 4$ computers (left) and $C = 16$ computers (right).

And so, even though CoCoA+ with SDCA updates should be a more powerful algorithm, its analysis [14] fails to show benefits over the simpler mini-batch SDCA, and out analysis here of mini-batch SDCA even dominates the CoCoA+ analysis. The reason for this is that CoCoA+ aims to be a more general framework capable of including arbitrary local solvers. Hence, necessarily, the analysis must be more conservative.

8 Numerical Experiments

In this Section we show that the cost of distribution is negligible (in terms of # iterations) when compared to standard mSDCA. We also show that if $b \gg 1$, then CoCoA+ is faster than mSDCA in practice. We have run experiments on 4 datasets (see Table 1). Note that most of the datasets are sparse (e.g, news20: an average tsample depends on 385 features out of 1.3M).

Standard vs. Distributed SDCA. Figure 1 (top row) compares standard and distributed SDCA. Recall that distributed sampling with $C = 1$ and standard mini-batch sampling coincide. On the x-axis is the parameter b and on the y-axis we plot how much more data-accesses we have to as b or C grow, to get achieve the same accuracy. We see that the lines are almost identical for various choices of C, which implies that the cost of using distributed mSDCA does not affect the number of iterations significantly. This is also supported by the theory (notice that in (12) we have $\beta_{\text{dist}}/\beta_{\text{std}} \approx 1$). Also note that, for news20 for instance, increasing b to 10^4 implies that the number of data-accesses (epochs) will increase by a factor of 11, which implies that # iterations will decrease almost by 1,000 for $b = 10^4$ when compared with $b = 1$.

mSDCA vs. CoCoA+. In Figure 1 (bottom row) we compare the mSDCA with CoCoA+ with SDCA as a local solver. We plot the duality gap as a function of epochs (if communication is negligible then the main cost is in computation) or iterations (if the communication cost is significant than this is the correct measure of performance). As the results suggest, is the communication cost it negligible then the mSDCA with small b is the best (as expected), however, if communications cost is significant, then CoCoA+ with large values of H significantly outperforms mSDCA.

References

[1] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[2] Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for l1-regularized loss minimization. *ICML*, 2011.

[3] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, pages 1647–1655, 2011.

[4] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13(1):165–202, 2012.

[5] Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takáč. Fast distributed coordinate descent for non-strongly convex losses. *arXiv:1405.5300*, 2014.

[6] Olivier Fercoq and Peter Richtárik. Accelerated, parallel and proximal coordinate descent. *arXiv:1312.5799*, 2013.

[7] C-J. Hsieh, K-W. Chang, C-J. Lin, S.S. Keerthi, and S. Sundarajan. A dual coordinate descent method for large-scale linear svm. In *ICML*, 2008.

[8] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *NIPS*, 2014.

[9] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *NIPS*, pages 315–323, 2013.

[10] Jakub Konečný, Jie Liu, Peter Richtárik, and Martin Takáč. mS2GD: Mini-batch semi-stochastic gradient descent in the proximal setting. *arXiv:1410.4744*, 2014.

[11] Jakub Konečný and Peter Richtárik. Semi-stochastic gradient descent methods. *arXiv:1312.1666*, 2013.

[12] Libsvm. *Datasets*. http://www.csie.ntu.edu.tw/~cjlin/ libsvmtools/datasets/binary.html.

[13] Zhaosong Lu and Lin Xiao. On the complexity analysis of randomized block-coordinate descent methods. *arXiv preprint arXiv:1305.4723*, 2013.

[14] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. *arXiv preprint arXiv:1502.03508*, 2015.

[15] Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *arXiv:1402.4419*, 2014.

[16] Jakub Mareček, Peter Richtárik, and Martin Takáč. Distributed block coordinate descent for minimizing partially separable functions. *arXiv:1406.0238*, 2014.

[17] Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

[18] Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling I: Algorithms and complexity. *arXiv preprint arXiv:1412.8060*, 2014.

[19] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *arXiv:1310.2059*, 2013.

[20] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *arXiv:1212.0873*, 2012.

[21] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 2013.

[22] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013.

[23] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *ArXiv:1209.1873*, 2012.

[24] Shai Shalev-Shwartz and Nathan Srebro. SVM optimization: inverse dependence on training set size. In *Proceedings of the 25th international conference on Machine learning*, pages 928–935. ACM, 2008.

[25] S.S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming: Series A and B*, pages 3–30, 2011.

[26] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. *Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing*, 2014.

[27] Martin Takáč, Avleen Singh Bijral, Peter Richtárik, and Nathan Srebro. Mini-batch primal and dual methods for SVMs. *ICML*, 2013.

[28] Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637, 2013.

[29] Tianbao Yang, Shenghuo Zhu, Rong Jin, and Yuanqing Lin. On theoretical analysis of distributed stochastic dual coordinate ascent. *arXiv preprint arXiv:1312.1031*, 2013.

[30] T. Zhang. Solving large scale linear prediction using stochastic gradient descent algorithms. In *ICML*, 2004.

A Technical Results

Lemma 4 (Lemma 2 in [23])**.** *For all $\alpha \in \mathbb{R}^n$:*

$$\mathcal{D}(\alpha) \leq \mathcal{P}(w^*) \leq \mathcal{P}(\mathbf{0}) \leq 1. \tag{14}$$

Moreover $\mathcal{D}(\mathbf{0}) \geq 0$.

Following Lemma is a minibatch extension of Lemma 1 in [23].

Lemma 5 (Expected increase of dual objective)**.** *Assume that ϕ_i^* is γ-strongly convex (γ can be also zero). Then, for any t and any $s \in [0,1]$ we have*

$$\mathbb{E}[\mathcal{D}(\alpha^{(t+1)}) - \mathcal{D}(\alpha^{(t)})] \geq b(\tfrac{s}{n}\mathcal{G}(\alpha^{(t)}) - (\tfrac{s}{n})^2 \tfrac{1}{2\lambda} G^{(t)}), \tag{15}$$

where

$$\begin{aligned} G^{(t)} &= \tfrac{1}{n}(\|u^{(t)} - \alpha^{(t)}\|_v^2 - \tfrac{\gamma\lambda n(1-s)}{s}\|u^{(t)} - \alpha^{(t)}\|^2) \\ &= \tfrac{1}{n}\textstyle\sum_{i=1}^n (v_i - \tfrac{\gamma\lambda n(1-s)}{s})(u_i^{(t)} - \alpha_i^{(t)})^2, \end{aligned} \tag{16}$$

$u_t = (u_1^{(t)}, \ldots, u_n^{(t)})^T$ and $-u_i^{(t)} \in \partial\phi_i(w_{\alpha^{(t)}}^T x_i)$.

Lemma 6 (Lemma 3 in [23])**.** *Let $\phi : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz continuous. Then for any $|\alpha| > L$ we have that $\phi^*(\alpha) = \infty$.*

Following lemma is a small extension of Lemma 4 in [23] to obtain more tide bounds in case each sample has different norm or when ESO bound is used. For example, in serial case we will have that $\forall t : G^t \leq 4L^2 \frac{\sum_{i=1}^n \|x_i\|^2}{n}$.

Lemma 7 (Bound on $G^{(t)}$)**.** *Suppose that for all i, ϕ_i is L-Lipschitz continuous. Then*

$$\forall t : G^{(t)} \leq 4L^2 \frac{\sum_{i=1}^n v_i}{n}. \tag{17}$$

Proof. Indeed,

$$G^{(t)} \overset{(16)}{=} \tfrac{1}{n}\textstyle\sum_{i=1}^n (v_i - \tfrac{\gamma\lambda n(1-s)}{s})(u_i^{(t)} - \alpha_i^{(t)})^2 \overset{\text{(Lemma 6)}}{\leq} \tfrac{1}{n}\textstyle\sum_{i=1}^n (v_i)(2L)^2. \qquad \square$$

Lemma 8 (Theorem 1 in [21])**.** *Fix $x_0 \in \mathbb{R}^N$ and let $\{x_k\}_{k\geq 0}$ be a sequence of random vectors in $\mathbb{R}^N$ with x_{k+1} depending on x_k only. Let $\phi : \mathbb{R}^N \to \mathbb{R}$ be a nonnegative function and define $\xi_k = \phi(x_k)$. Lastly, choose accuracy level $0 < \epsilon < \xi_0$, confidence level $0 < \rho < 1$, and assume that the sequence of random variables $\{\xi_k\}_{k\geq 0}$ is nonincreasing and has one of the following properties:*

- *(i) $\mathbb{E}[\xi_{k+1} \mid x_k] \leq (1 - \frac{\xi_k}{c_1})\xi_k$, for all k, where $c_1 > \epsilon$ is a constant,*
- *(ii) $\mathbb{E}[\xi_{k+1} \mid x_k] \leq (1 - \frac{1}{c_2})\xi_k$, for all k such that $\xi_k \geq \epsilon$, where $c_2 > 1$ is a constant.*

If property (i) holds and we choose $K \geq 2 + \frac{c_1}{\epsilon}(1 - \frac{\epsilon}{\xi_0} + \log(\frac{1}{\rho}))$, or if property (ii) holds, and we choose $K \geq c_2 \log(\frac{\xi_0}{\epsilon\rho})$, then $\mathbb{P}(\xi_K \leq \epsilon) \geq 1 - \rho$.

B Proofs

B.1 Proof of Lemma 5

Let us define $\mathbf{T}_\alpha$ as an unique maximizer of a function $\mathcal{H}(t, \alpha)$ defined in (7), i.e.

$$\mathbf{T}_\alpha := \arg\max_t \mathcal{H}(t, \alpha). \tag{18}$$

Let us now state some basic properties about function $\mathcal{H}$. We have that $\forall t, \alpha \in \mathbb{R}^n$ and sampling $\hat{S}$:

- $\mathcal{H}(\mathbf{0}, \alpha) = \mathcal{D}(\alpha)$,

- from ESO we have

$$\mathbb{E}[\mathcal{D}(\alpha + t_{[\hat{S}]})] \geq (1 - \tfrac{b}{n})\mathcal{D}(\alpha) + \tfrac{b}{n}\mathcal{H}(t, \alpha), \tag{19}$$

- $\mathcal{H}(t, \alpha) \leq \mathcal{H}(\mathbf{T}_\alpha, \alpha)$.

Convex conjugate maximal property implies that

$$\phi_i^*(-u_i^{(t)}) = -u_i^{(t)} w_{\alpha^{(t)}}^T x_i - \phi_i(w_{\alpha^{(t)}}^T x_i). \tag{20}$$

Let us estimate the expected change of dual objective.

$$\begin{aligned}
\frac{n}{b}\mathbb{E}[\mathcal{D}(\alpha^{(t)}) - \mathcal{D}(\alpha^{(t+1)})] &= \frac{n}{b}\mathbb{E}[\mathcal{D}(\alpha^{(t)}) - \mathcal{D}(\alpha^{(t)} + (\mathbf{T}_{\alpha^{(t)}})_{[\hat{S}]})] \overset{(19)}{\leq} \mathcal{D}(\alpha^{(t)}) - \mathcal{H}(\mathbf{T}_{\alpha^{(t)}}, \alpha^{(t)}) \\
&= \frac{1}{n}\sum_{i=1}^n \left(\phi_i^*(-(\alpha_i + (\mathbf{T}_{\alpha^{(t)}})^{(i)})) - \phi_i^*(-\alpha_i^{(t)})\right) \\
&\quad + \frac{\lambda}{2}\left(\left\|\frac{1}{\lambda n}\mathbf{T}_{\alpha^{(t)}}\right\|_v^2 + 2\left(\frac{1}{\lambda n}\mathbf{T}_{\alpha^{(t)}}\right)^T X w_\alpha\right) \\
&\leq \frac{1}{n}\sum_{i=1}^n \left(\phi_i^*(-(\alpha_i^{(t)} + s(u_i - \alpha_i^{(t)})) - \phi_i^*(-\alpha_i^{(t)})\right) \\
&\quad + \frac{\lambda}{2}\left(\left\|\tfrac{1}{\lambda n}s(u - \alpha^{(t)})\right\|_v^2 + 2\left(\tfrac{1}{\lambda n}s(u - \alpha^{(t)})\right)^T X w_\alpha\right).
\end{aligned}$$

Using γ-strong convexity of ϕ_i^* we have that

$$\phi_i^*(-(\alpha_i^{(t)} + s(u_i - \alpha_i^{(t)})) \leq s\phi_i^*(-u_i) + (1 - s)\phi_i^*(-\alpha_i^{(t)}) - \tfrac{\gamma}{2}(1 - s)s(u_i - \alpha_i^{(t)})^2. \tag{21}$$

Therefore,

$$\begin{aligned}
\frac{n}{b}\mathbb{E}[\mathcal{D}(\alpha^{(t)}) - \mathcal{D}(\alpha^{(t+1)})] &\overset{(21)}{\leq} \frac{1}{n}\sum_{i=1}^n \left(s\phi_i^*(-u_i) + s u_i x_i^T w_{\alpha^{(t)}} - s\phi_i^*(-\alpha_i^{(t)}) - \tfrac{\gamma}{2}(1 - s)s(u_i - \alpha_i^{(t)})^2)\right) \\
&\quad + \frac{\lambda}{2}\left(\left\|\tfrac{1}{\lambda n}s(u - \alpha^{(t)})\right\|_v^2 + 2\left(\tfrac{1}{\lambda n}s(-\alpha^{(t)})\right)^T X w_{\alpha^{(t)}}\right) \\
&\overset{(20)}{\leq} \frac{s}{n}\sum_{i=1}^n \left(-u_i w_{\alpha^{(t)}}^T x_i - \phi_i(w_{\alpha^{(t)}}^T x_i) + u_i x_i^T w_{\alpha^{(t)}} - \phi_i^*(-\alpha_i^{(t)}))\right) \\
&\quad + \frac{\lambda}{2}\left(-\frac{\gamma}{\lambda n}(1 - s)s\|u - \alpha\|^2 + \left\|\tfrac{1}{\lambda n}s(u - \alpha^{(t)})\right\|_v^2 + 2\left(\tfrac{1}{\lambda n}s(-\alpha^{(t)})\right)^T X w_{\alpha^{(t)}}\right).
\end{aligned}$$

Substituting the definition of duality gap (G) we obtain

$$\begin{aligned}
\frac{n}{b}\mathbb{E}[\mathcal{D}(\alpha^{(t)}) - \mathcal{D}(\alpha^{(t+1)})] &\leq \frac{s}{n}\sum_{i=1}^n \left(-\phi_i(w_{\alpha^{(t)}}^T x_i) - \phi_i^*(-\alpha_i^{(t)}) - \alpha_i^{(t)} w_\alpha^T x_i)\right) \\
&\quad + \frac{\lambda}{2}\left(-\frac{\gamma}{\lambda n}(1 - s)s\|u - \alpha\|^2 + \left\|\tfrac{1}{\lambda n}s(u - \alpha^{(t)})\right\|_v^2\right) \\
&= -s\mathcal{G}(\alpha^{(t)}) + \frac{\lambda}{2}\left(\left\|\tfrac{1}{\lambda n}s(u - \alpha^{(t)})\right\|_v^2 - \frac{\gamma}{\lambda n}(1 - s)s\|u - \alpha\|^2\right). \\
&= -s\mathcal{G}(\alpha^{(t)}) + \frac{1}{2\lambda}\left(\frac{s}{n}\right)^2\left(\left\|(u - \alpha^{(t)})\right\|_v^2 - \frac{\gamma n\lambda(1 - s)}{s}\|u - \alpha\|^2\right).
\end{aligned}$$

Multiplying both sides by $-\frac{b}{n}$ we obtain (15).

B.2 Proof of Theorem 3

At first let us estimate expected change of dual feasibility.

$$\begin{aligned}\mathbb{E}[\epsilon_D^{(t+1)}] &\overset{(15)}{=} \mathbb{E}[\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t+1)})] = \mathbb{E}[\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t+1)}) + \mathcal{D}(\alpha^{(t)}) - \mathcal{D}(\alpha^{(t)})] \\ &= \mathbb{E}[\mathcal{D}(\alpha^{(t)}) - \mathcal{D}(\alpha^{(t+1)}) + \epsilon_D^{(t)}] \overset{(15),(17)}{\leq} -b\left(\frac{s}{n}\mathcal{G}(\alpha^{(t)}) - \left(\frac{s}{n}\right)^2 \frac{1}{2\lambda}G\right) + \mathbb{E}[\epsilon_D^{(t)}] \\ &\leq -b\frac{s}{n}\mathbb{E}[\epsilon_D^{(t)}] + b\left(\frac{s}{n}\right)^2 \frac{1}{2\lambda}G + \mathbb{E}[\epsilon_D^{(t)}] = (1 - b\tfrac{s}{n})\mathbb{E}[\epsilon_D^{(t)}] + b\left(\frac{s}{n}\right)^2 \frac{}{2\lambda}G. \end{aligned} \tag{22}$$

From the above follows that

$$\mathbb{E}[\epsilon_D^{(t)}] \leq (1 - b\tfrac{s}{n})^t \epsilon_D^0 + b\left(\frac{s}{n}\right)^2 \frac{}{2\lambda}G \sum_{i=0}^{t-1}(1 - b\tfrac{s}{n})^i \leq (1 - b\tfrac{s}{n})^t \epsilon_D^0 + \left(\frac{s}{n}\right)\frac{G}{2\lambda}. \tag{23}$$

Choice of $s = 1$ and $t = t_0 := \max\{0, \lceil \frac{n}{b}\log(2\lambda n \epsilon_D^{(0)}/(G))\rceil\}$ will lead to

$$\mathbb{E}[\epsilon_D^{t_0}] \leq (1 - \tfrac{b}{n})^{t_0}\epsilon_D^{(0)} + \frac{s}{n}\frac{G}{2\lambda} \leq \frac{G}{2\lambda n \epsilon_D^{(0)}}\epsilon_D^{(0)} + \frac{1}{n}\frac{G}{2\lambda} = \frac{G}{\lambda n}. \tag{24}$$

Following the proof in [23] we are now going to show that

$$\forall t \geq t_0 : \mathbb{E}[\epsilon_D^{(t)}] \leq \frac{2G}{\lambda(2n + b(t - t_0))}. \tag{25}$$

Clearly, (24) implies that (25) holds for $t = t_0$. Now imagine that it holds for any $t \geq t_0$ then we show that it also has to hold for $t + 1$. Indeed, using $s = \frac{2n}{2n + b(t-t_0)}$ we obtain

$$\begin{aligned}\mathbb{E}[\epsilon_D^{(t+1)}] &\overset{(22)}{\leq} (1 - b\tfrac{s}{n})\mathbb{E}[\epsilon_D^{(t)}] + b\left(\frac{s}{n}\right)^2 \frac{1}{2\lambda}G \\ &\overset{(25)}{\leq} (1 - b\tfrac{s}{n})\frac{2G}{\lambda(2n + b(t - t_0))} + b\left(\frac{s}{n}\right)^2 \frac{1}{2\lambda}G \\ &= (1 - b\frac{2}{2n + b(t - t_0)})\frac{2G}{\lambda(2n + b(t - t_0))} + b\left(\frac{2}{2n + b(t - t_0)}\right)^2 \frac{1}{2\lambda}G \\ &= \frac{2G}{\lambda}\left(\frac{1}{2n + b(t - t_0) + b}\right)\left(\frac{2n + b(t - t_0) + b}{1}\right)\left(\frac{2n + b(t - t_0) - b}{(2n + b(t - t_0))^2}\right) \\ &= \frac{2G}{\lambda(2n + b(t - t_0) + b)}\frac{(2n + b(t - t_0) + b)(2n + b(t - t_0) - b)}{(2n + b(t - t_0))^2} \\ &\leq \frac{2G}{\lambda(2n + b(t - t_0) + b)}. \end{aligned} \tag{26}$$

In the last inequality we have used the fact that geometric mean is less or equal to arithmetic mean. If $\bar{\alpha}$ is defined as (9) then we obtain that

$$\begin{aligned}\mathbb{E}[\mathcal{G}(\bar{\alpha})] = \mathbb{E}\left[\mathcal{G}\left(\sum_{t=T_0}^{T-1}\tfrac{1}{T-T_0}\alpha^{(t)}\right)\right] &\leq \tfrac{1}{T-T_0}\mathbb{E}\left[\sum_{t=T_0}^{T-1}\mathcal{G}\left(\alpha^{(t)}\right)\right] \\ &\overset{(15)}{\leq} \tfrac{1}{T-T_0}\mathbb{E}\left[\sum_{t=T_0}^{T-1}\left(-\frac{n}{s}\frac{1}{b}\mathbb{E}[\mathcal{D}(\alpha^{(t)}) - \mathcal{D}(\alpha^{(t+1)})] + \left(\frac{s}{n}\right)\frac{1}{2\lambda}\left(\tfrac{1}{n}\left\|u_t - \alpha^{(t)}\right\|_v^2\right)\right)\right] \\ &\overset{(17)}{\leq} \frac{n}{s}\frac{1}{b}\frac{1}{T - T_0}\left(\mathbb{E}[\mathcal{D}(\alpha^{(T)})] - \mathbb{E}[\mathcal{D}(\alpha^{(T_0)})]\right) + \frac{s}{n}\frac{G}{2\lambda} \\ &\leq \frac{n}{s}\frac{1}{b}\frac{1}{T - T_0}\left(\mathcal{D}(\alpha^*) - \mathbb{E}[\mathcal{D}(\alpha^{(T_0)})]\right) + \frac{s}{n}\frac{G}{2\lambda}. \end{aligned} \tag{27}$$

Now, if $T \geq \lceil \frac{n}{b}\rceil + T_0$ such that $T_0 \geq t_0$ we obtain

$$\begin{aligned}\mathbb{E}[\mathcal{G}(\bar{\alpha})] &\overset{(25)}{\leq} \frac{n}{s}\frac{1}{b}\tfrac{1}{T-T_0}\left(\frac{2G}{\lambda(2n + b(T_0 - t_0))}\right) + \frac{s}{n}\frac{G}{2\lambda} \\ &= \frac{G}{\lambda}\left(\frac{n}{s}\frac{1}{b(T - T_0)}\left(\frac{2}{(2n + b(T_0 - t_0))}\right) + \frac{s}{2n}\right). \end{aligned}$$

Using $s = \frac{n}{b(T-T_0)}$ we obtain that

$$\mathbb{E}[\mathcal{G}(\bar{\alpha})] \leq \frac{G}{b\lambda}\left(\frac{2}{2\frac{n}{b} + (T_0 - t_0)} + \frac{1}{2(T-T_0)}\right).$$

To have this quantity $\leq \epsilon_{\mathcal{G}}$ we obtain that T, t_0, T_0 has to satisfy (10). The fact that $T_0 \geq t_0 + \frac{1}{b}\left(\frac{4G}{\lambda\epsilon_{\mathcal{G}}} - 2n\right)_+$ implies that right-hand site of (26) is $\leq \epsilon_{\mathcal{G}}$.

B.3 Proof of Theorem 2

If function ϕ_i is $(1/\gamma)$-smooth then ϕ_i^* is γ-strongly convex. If we plug $s = \tilde{s} = \frac{\lambda n\gamma}{\|v\|_\infty + \lambda n\gamma} \in (0,1)$ into (16) we obtain that $\forall t : G^{(t)} \leq 0$. Hence (15) will read as follows

$$\mathbb{E}[\mathcal{D}(\alpha^{(t+1)}) - \mathcal{D}(\alpha^{(t)})] \geq b\frac{\tilde{s}}{n}\mathcal{G}(\alpha^{(t)}) = b\frac{\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\mathcal{G}(\alpha^{(t)}) \geq b\frac{\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}(\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t)})). \tag{28}$$

Using the fact that $\mathbb{E}[\mathcal{D}(\alpha^{(t+1)}) - \mathcal{D}(\alpha^{(t)})] = \mathbb{E}[\mathcal{D}(\alpha^{(t+1)}) - \mathcal{D}(\alpha^*)] + \mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t)})$ we have

$$\mathbb{E}[\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t+1)})] \leq \left(1 - b\frac{\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\right)(\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t)})). \tag{29}$$

Therefore if we denote by $\epsilon_D^{(t)} = \mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t)})$ we have that

$$\mathbb{E}[\epsilon_D^{(t)}] \leq \left(1 - b\frac{\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\right)^t \epsilon_D^{(0)} \overset{(14)}{\leq} \left(1 - b\frac{\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\right)^t \leq \exp\left(-bt\frac{\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\right).$$

Right hand site will be smaller than some ϵ_D if

$$t \geq \frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\log\frac{1}{\epsilon_D}.$$

Moreover, to bound the duality gap we have

$$b\frac{\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\mathcal{G}(\alpha^{(t)}) \overset{(28)}{\leq} \mathbb{E}[\epsilon_D^{(t)} - \epsilon_D^{(t+1)}] \leq \epsilon_D^{(t)}. \tag{30}$$

Therefore $\mathcal{G}(\alpha^{(t)}) \leq \frac{\|v\|_\infty + \lambda n\gamma}{b\lambda\gamma}\epsilon_D^{(t)}$. Hence if $\epsilon_D \leq \frac{b\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\epsilon_{\mathcal{G}}$ then $\mathcal{G}(\alpha^{(t)}) \leq \epsilon_{\mathcal{G}}$. Therefore after

$$t \geq \frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\log\left(\frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\frac{1}{\epsilon_{\mathcal{G}}}\right).$$

iterations we have duality gap less than $\epsilon_{\mathcal{G}}$ and the first part of the proof is done. To show the second part of Theorem let us sum (30) over $t = T_0, \ldots, T-1$ to obtain

$$\mathbb{E}\left[\frac{1}{T-T_0}\sum_{t=T_0}^{T-1}\mathcal{G}(\alpha^{(t)})\right] \leq \frac{\|v\|_\infty + \lambda n\gamma}{b\lambda\gamma}\frac{1}{T-T_0}\mathbb{E}[\mathcal{D}(\alpha^{(T)}) - \mathcal{D}(\alpha^{(T_0)})]. \tag{31}$$

Now, if we choose $\bar{w}, \bar{\alpha}$ to be either average vectors or a randomly chosen vector over $t \in \{T_0 + 1, \ldots, T\}$, then we have

$$\mathbb{E}[\mathcal{G}(\bar{\alpha})] \overset{(31)}{\leq} \frac{\|v\|_\infty + \lambda n\gamma}{b\lambda\gamma}\frac{1}{T-T_0}\mathbb{E}[\mathcal{D}(\alpha^{(T)}) - \mathcal{D}(\alpha^{(T_0)})] \leq \frac{\|v\|_\infty + \lambda n\gamma}{b\lambda\gamma}\frac{1}{T-T_0}\mathbb{E}[\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(T_0)})].$$

Hence to have $\mathbb{E}[\mathcal{G}(\bar{\alpha})] \leq \epsilon_{\mathcal{G}}$ it is sufficient to choose

$$\mathbb{E}[\epsilon_D^{(T_0)}] \leq \frac{b\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}(T-T_0)\epsilon_{\mathcal{G}}.$$

Therefore we need T_0 to satisfy

$$T_0 \geq \frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\log\left(\frac{\|v\|_\infty}{b}\left(\frac{1}{\lambda\gamma} + \frac{n}{\|v\|_\infty}\right)\frac{1}{(T-T_0)\epsilon_{\mathcal{G}}}\right).$$

To get a high probability result we use Lemma 8 with $\xi^{(t)} = \mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(t)})$, $c_2 = \frac{\|v\|_\infty + \lambda n \gamma}{b\lambda\gamma}$ (see (29)) and $\epsilon = \frac{b\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\epsilon_{\mathcal{G}}$ to obtain that after

$$\tilde{T} = c_2 \log\left(\frac{\xi^{(0)}}{\epsilon\rho}\right) \overset{\text{Lemma 4}}{\leq} \frac{\|v\|_\infty + \lambda n\gamma}{b\lambda\gamma} \log\left(\frac{1}{\epsilon\rho}\right)$$

$$1 - \rho \leq \mathbb{P}\left(\mathcal{D}(\alpha^*) - \mathcal{D}(\alpha^{(\tilde{T})}) \leq \epsilon\right) \overset{(30)}{\leq} \mathbb{P}\left(\frac{b\lambda\gamma}{\|v\|_\infty + \lambda n\gamma}\mathcal{G}(\alpha^{(t)}) \leq \epsilon\right) = \mathbb{P}\left(\mathcal{G}(\alpha^{(t)}) \leq \epsilon_{\mathcal{G}}\right).$$